Inteligência Computacional Aplicada à Resolução do Problema do Corte Unidimensional

Tiago Zonta[1], Luiz F. J. Maia[2], Ilaim Costa Junior[3], Itamar Leite de Oliveria[4]

[1]Faculdade Exponencial FIE.
tiagoz@gmail.com

[2]UFSC – Universidade Federal de Santa Catarina.
maia@inf.ufsc.br

[3]UDESC – Universidade do Estado de Santa Catarina.
ilaim@joinville.udesc.br

[4]UFSC – Universidade Federal de Santa Catarina.
itamar@intelab.ufsc.br

Resumo - Neste trabalho é feito um estudo para resolução do Problema do Corte Unidimensional (PCU) utilizando técnicas de Programação Matemática e Inteligência Computacional (IC). No PCU, vários objetos de dimensão **L**, devem ser cortados em **n** itens menores com dimensões $\mathbf{l_i}$ e valor de utilidade $\mathbf{v_i}$. O problema é achar a melhor forma de se fazer este corte de modo a minimizar a perda dos objetos de comprimento **L** a serem cortados e maximizar a soma dos valores de utilidade de cada um dos itens menores de comprimento $\mathbf{l_i}$. Foram utilizadas quatro técnicas: Algoritmo First Fit Decreasing (FFD), Algoritmo Guloso (AGU), Limitante de Dantzig (LD) e Algoritmos Genéticos (AG).
Palavras-chave: Otimização, Inteligência Computacional, Problema do Corte Unidimensional.

Abstract - In this work a study is done towards the One-dimensional Cutting Stock Problem (PCU)'s resolution by using Mathematical Programming and Artificial Intelligence (AI)'s techniques. In PCU problem, some objects of **L** dimension must be cut in **n** shorter items with $\mathbf{l_i}$ dimensions and an utility value $\mathbf{v_i}$. The problem is to find the best form of doing this cut in order to minimize the loss of the objects to be cut and to maximize the sum of the used item's utility values. Four techniques have been used: First Fit Decreasing (FFD), Greed Algorithm (AGU), Dantzig Algorithm (LD) and Genetic Algorithms (AG).
Key-words: Optimization, Computational Intelligence, One-dimensional Cutting Stock Problem.

1. Introdução

As pesquisas sobre modelos computacionais inteligentes têm, nos últimos anos, caracterizado-se pela tendência em buscar inspiração na natureza, onde existem inúmeros exemplos vivos de processos que podem ser ditos "inteligentes". Embora não se possa afirmar que soluções tiradas destes processos sejam todas "ótimas", não há a menor dúvida de que os processos naturais, em particular os relacionados diretamente com os seres vivos, são bem concebidos e adequados ao nosso mundo [11]. Um destes modelos inspirado na natureza é conhecido como algoritmo genético [2][11].

Neste sentido será apresentada uma solução para o PCU utilizando AG [1][2] e depois será feita uma comparação com algoritmos exatos – FFD, AGU e LD [9].

Inicialmente apresentar-se-á a motivação e justificativa do porquê foi realizado o trabalho, uma introdução ao PCU, bem como seu modelo matemático; Posteriormente um estudo resumido sobre as técnicas utilizadas; e por fim resultados comparativos das soluções obtidas com cada uma delas.

2. Motivação e justificativa

A motivação e justificativa para resolução do PCU surgiram com a necessidade de se aplicar uma solução inteligente para um problema comercial encontrado numa empresa de montagem de parede de divisória. Essas paredes são bastante

conhecidas por dividir ambientes de forma econômica e rápida, mas, com um tempo de desenvolvimento de projetos (envolvendo desenho das paredes, calculos da quantidade de materiais e orçamento) realizados praticamente manualmente e cortados na experiência conhecida como “no olho”.

Por apresentarem padrões nos tamanhos das placas, chamadas painéis, foi identificada a possível aplicação de métodos da pesquisa operacional conhecidas como problema do corte e a possível aplicação de métodos da Inteligência computacional que aplicam computação evolutiva [8][10].

O problema do corte pode ser encontrado na literatura na forma de uni, bi ou tridimensional [3][5][6][7][9]. No corte dos painéis foi identificada a possibilidade da aplicação da forma bidimensional. A forma unidimensional é o objetivo de estudo deste trabalho e é responsável por grande parte no problema do corte de um projeto de paredes divisórias.

Todo o processo é realizado após o desenho das paredes, aplicando os tamanhos padrões dos painéis e obtendo os itens a serem produzidos com a aplicação do corte. Com essa ferramenta será possível desenhar e determinar todo o material e orçamento de uma obra.

3. Problema do Corte Unidimensional

Suponha que um objeto deva ser cortado ao longo de seu comprimento em itens (pedaços) de comprimentos especificados. Cada item possui um valor associado que chamamos de valor de utilidade. Itens cujos comprimentos não foram especificados são considerados perdas e têm valores de utilidade nulos. Surge então um problema de otimização combinatória [9][14]:

Maximizar VALOR DE UTILIDADE TOTAL OU **Minimizar** AS PERDAS.

O problema é achar a melhor forma de cortar o objeto para produzir os itens, de modo que, o valor de utilidade total seja máximo (equivale a dizer que a perda de material seja mínima).

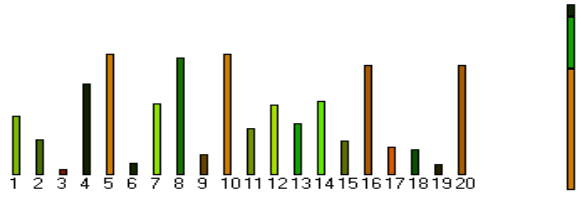

Figura 1 – Exemplo do PCU, 20 itens (a) devem ser cortados para produzir uma barra (b). Uma solução foi produzida utilizando-se os pedaços 5,13 e 19 [13].

Na Figura 1 é apresentado um exemplo deste problema. Neste exemplo pode-se observar que ainda sobraram itens para serem utilizados. Este problema, então, pode ser visualizado considerando um estoque, ou seja, determinado número de objetos que devem ser cortados para produção de itens menores necessário para suprir um estoque, ou a utilização de um estoque de itens que deverão produzir o menor número possível de objetos. Na figura 2 é apresentado uma extensão deste problema, pode-se visualizar a diferença e o aumento da complexidade ao estender o problema a múltiplos objetos a serem cortados.

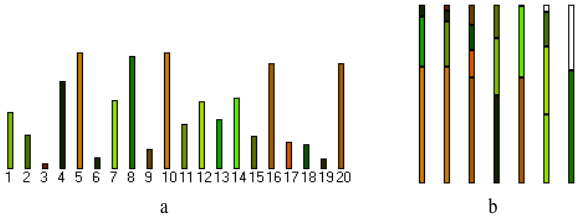

Figura 2 –Exempo de um problema com 20 ítens (a) que devem produzir barras (b) com a menor perda possível. Foram produzidas 7 barras completas, das quais 3 tiveram perdas (retângulo branco no topo das três útimas barras em b [13].

Observa-se nas figuras que o corte é feito em apenas uma dimensão do objeto, ou seja, o objeto é cortado apenas na horizontal. Problemas com esta característica são chamados **Problemas de Corte Unidimensional (PCU)**. Se houverem cortes em ambas dimensões, horizontal e vertical, tem-se um problema de corte bidimensional.

O problema fica bem entendido a partir de uma análise de seu modelo matemático, o qual é apresentado abaixo:

Dados do problema:

***m*:** número de tipos de itens;
***v*$_i$:** valor de utilidade do item tipo ***i(i>0), i =1, ..., m*;**
***l*$_i$:** comprimento do item tipo ***i(i>0), i =1, ..., m*;**
***L*:** restrição física do problema (comprimento da barra).

Variáveis de decisão:
x*$_i$:** item ***i (1 ou 0) faz parte da solução, ***i =1, ..., m.***

$$maximizar \quad z = \sum_{i=1}^{m} v_i x_i \qquad (1.1)$$

$$sujeito\ a:$$

$$\sum_{i=1}^{m} l_i x_i \leq L \qquad (1.2)$$

$$x_i \geq 0 \ e \ inteiro, i = 1,\ldots,m. \qquad (1.3)$$

Modelo 1 – modelo genérico do PCU [9].

Observe que o modelo 1 possui uma restrição, chamada restrição física (1.2). Esta restrição diz que o somatório dos itens utilizados não deve exceder ao tamanho do objeto.

O foco deste trabalho foi resolver o PCU para múltiplos objetos de comprimento **L**. A seguir apresenta-se o modelo matemático para este tipo de problema.

Dados do problema:

m: número de objetos;
n: número de itens;
lj: comprimento do item tipo *j(>0), j(<=L), j =1, ..., m*;
L (>0): restrição física do problema (comprimento das barras e comprimento máximo dos itens).
Variáveis de decisão:
xij: item *j* utilizado pelo objeto *i (0 ou 1)*;

$$\text{maximizar} \quad z = \sum_{i=1}^{m} \sum_{j=1}^{n} l_j x_{ij} \qquad (2.1)$$

$$sujeito \quad a \quad : \sum_{i=1}^{m} x_{ij} \leq 1,\ j = 1,...,\ n, \qquad (2.2)$$

$$\sum_{j=1}^{n} l_j x_{ij} \leq L, i = 1,...,\ m, \qquad (2.3)$$

$$x_{ij} = 0\ ou\ 1, i = 1,...,\ m.\ j = 1,...,\ n. \qquad (2.4)$$

$$l_j > 0, l_j \leq L, L > 0 \qquad (2.5)$$

Modelo 2 – Problema do corte 0 ou 1, para múltiplos objetos em estoque [3].

4. Algoritmos Exatos para a Solução do PCU

A partir deste ponto, será feita uma apresentação de cada um dos algoritmos utilizados neste trabalho para a resolução do PCU.

4.1 Algoritmo First Fit Decreasing (FFD)

O algoritmo FFD, já em sua forma original, resolve o problema do corte para múltiplos objetos em estoque, isto é, resolve o PCU para diversos objetos. O funcionamento deste algoritmo baseia-se na idéia de que o primeiro item não utilizado e útil para a solução corrente, se puder ser encaixado nela, deverá ser utilizado [15]. O algoritmo FFD utiliza apenas a restrição física do problema, não fazendo mais nenhuma restrição aos itens que serão utilizados na solução.

Pode-se supor o problema sendo resolvido pelo FFD considerando cinco itens de tamanhos menores que devem ser utilizados para criar barras de 2 metros. Veja na tabela 1 o resultado mostrando que o único trabalho do algoritmo é o de verificar se o item pode ser utilizado em alguma das soluções já existentes. Caso não possa ser utilizado, é aberta uma nova solução.

Itens	Itens ordenados	Rodadas	Itens utilizados
1,00 m	1,50 m	1	1
0,50 m	1,25 m	1	2
0,75 m	1,00 m	1	3
1,25 m	0,75 m	1	2
1,50 m	0,50 m	1	1

Tabela 1 – Resultados utilizando FFD. São cinco itens utilizando três barras de 2 metros [13].

4.2 Algoritmo Guloso (AGU)

Em seu funcionamento, pode-se ver o porquê de seu sugestivo nome. A cada iteração abocanha o primeiro item viável seguindo uma ordem dada, sem se preocupar com o que vai acontecer com a solução apontada como viável. Isso quer dizer que o algoritmo apresentará uma única solução que não se pode afirmar que seja ótima.

O funcionamento do AGU apresentado em [9][15] faz uso da mesma idéia do FFD, o primeiro item não utilizado e útil para solução será utilizado. Ele possui apenas uma diferença em relação à sua forma original, resolve o PCU apenas para um objeto em estoque, isto é, aplica o corte em apenas um objeto.

A tabela 2 mostra um exemplo do problema do corte sendo resolvido pelo AGU considerando cinco itens de tamanhos menores que devem ser utilizados para criar barras de 2 metros. O algoritmo deverá ser repetido até não existir nenhum item sobrando. Veja na tabela abaixo o resultado tal que o único trabalho do algoritmo é o de verificar se o item já foi utilizado em outra rodada e encaixa-se na resolução corrente.

Itens	Itens ordenados	Rodadas	Itens utilizados
1,0 m	1,50 m	1	1
0,50 m	1,25 m	2	2
0,75 m	1,0 m	3	3
1,25 m	0,75 m	2	2
1,50 m	0,50 m	1	1

Tabela 2 – Resultados utilizando AGU. São cinco itens, utilizando 3 barras de 2 metros. Com perda de 1m na terceira barra [13].

4.3 Algoritmo usando Limitante de Dantzig (LD)

Como pode ser visto em [9], o LD tem o seguinte mecanismo de funcionamento:

Forward move: Inserção da maior quantidade possível (inteira) de itens consecutivos na solução que está sendo investigada (Solução Corrente);

Backtracking move: remoção do último item inserido na solução corrente, realizando uma nova procura por um item de melhor utilidade. O *Backtracking move* é a chamada mais importante do algoritmo. É nela que se aplica o retorno à exploração do resultado;

Limitante Superior: Quando um determinado item não puder ser agregado à solução corrente, calcula-se o limitante associado. Faz-se, então, uma comparação desta última com a melhor solução encontrada até então. Se não for potencialmente possível melhorar a melhor solução, faz-se um *Backtracking*, caso contrário um f*orward move* ocorre;

Testes de parada: são necessários dois testes de parada: Término parcial – se o último item tiver sido considerado, testa-se a **solução corrente** com a melhor solução, para uma possível atualização. Término geral – se não for possível fazer *Backtracking* algum.

Considerando os itens da tabela 3 e a situação já citada nos exemplos do FFD e AGU é possível descrever uma possível forma de funcionamento.

Na primeira rodada do algoritmo será encaixado o máximo possível de itens menores. Neste caso seria utilizado o item 2,00m, sendo, então, verificado se o resultado já satisfaz a condição física do problema. Como o resultado satisfaz a condição, é feito um armazenamento da solução e realizado um *Backtracking* que retirará o item da solução corrente e reiniciará-la neste caso, como temos apenas um item, a solução corrente ficará vazia. Então, ainda na primeira rodada, a solução corrente está sem nenhum item e é encaixado 1,50m, encaixa também o 1,00m, mas retira porque extrapola a condição física, da mesma forma ocorre com os dois 0,75m, inserindo assim o 0,50 completando os 2m da solução. Novamente são armazenados os itens e executado o *Backtracking,* que retirar o 0,50 iniciando a procura encontrando então (1,50 + 0,25 + 0,25).

Baseado neste exemplo, pode-se ver que tem-se três possíveis soluções na primeira rodada do algoritmo. Neste caso, então, poderiam ser utilizadas outras restrições, como, por exemplo, escolher a solução que possui maior número de itens, restrição esta que será utilizada para demonstrar um possível resultado utilizando os itens menores da tabela 3. A coluna **rodadas** identifica os itens que participaram da seleção, e a coluna itens utilizados demonstra em qual rodada eles foram escolhidos como solução viável.

Itens	Itens ordenados	Rodadas	Itens utilizados
1,00 m	2,00 m	1,2,3	3
0,25 m	1,50 m	1	1
0,75 m	1,00 m	2,3,4	4
0,50 m	0,75 m	2	2
0,75 m	0,75 m	2	2
0,25 m	0,50 m	1,2	2
1,50 m	0,25 m	1	1
2,00 m	0,25 m	1	1

Tabela 3 – Resultados utilizando LD. São sete itens, utilizando 4 barras de 2 metros. Com perda de 1m na quarta barra.

Todas as tabelas mostradas com os exemplos de execução dos algoritmos, com exceção do LD que utiliza o *Backtracking* para tentar melhorar a solução, ao encontrar uma solução tida como "viável", considerando os critérios do problema, dão a busca como satisfeita. Isso é encontrado na literatura como busca por um valor maximal [12].

4.4 Algoritmos Genéticos (AG)

Muitos dos problemas que a Inteligência Computacional tenta resolver tratam da busca de uma solução num espaço vasto de candidatos sujeitos a restrições do tempo. Quando não há nenhum conhecimento sobre prioridade da busca no problema não se deve utilizar estratégias específicas do domínio. Com a proposta de resolver problemas de busca adaptativa, surgem os algoritmos genéticos, (AG) [2][4] que são inspirados no processo genético e evolutivo dos organismos vivos, nos quais o conhecimento para controlar a busca é obtido dinamicamente [11].

As buscas e as otimizações tradicionais são inicializadas a partir de um único candidato que, iterativamente, é controlado pela utilização de algumas heurísticas diretamente ligadas ao problema a ser resolvido. Os processos heurísticos são, geralmente, não algorítmicos e sua emulação em computadores pode ser muito complexa.

Seguindo essa idéia foi determinado que seria possível trocar as resoluções, até então utilizadas no sistema de projeto de paredes que aplicam essa busca, por otimização por uma busca evolutiva que pudesse aplicar uma busca dinâmica para o problema proposto.

4.4.1 Representação do cromossomo

Um dos primeiros requisitos a ser definido para utilização de AG's na resolução de um determinado problema é a determinação de uma representação para o indivíduo, ou seja, para o cromossomo. Cada cromossomo representará uma solução para o problema, sendo que, a partir de uma solução pré-inicializada, será realizado todo o processo de evolução do algoritmo. Processo este realizado pelos operadores genéticos.

Existem diversas formas para representação de um cromossomo. Os AG's, na sua forma original (AG's convencionais), trabalham normalmente com uma representação de strings de bits ou caracteres. Esta é uma representação binária ([0,1] – veja Figura 3) conhecida como problema binário. Segundo os modelos 1 e 2 já vistos, esta representação se torna ideal para o problema proposto, no qual o cromossomo 1 determinará se o gene faz parte e 0 se não faz parte da solução.

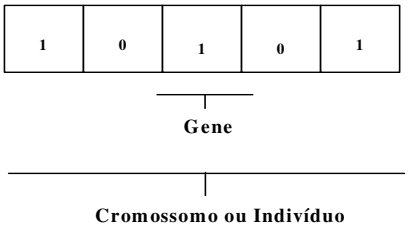

Figura 3 - Representação de um cromossomo binário.

Com a representação da Figura 3, e com o uso de operações de manipulação de bits, é possível definir as funções de seleção, cruzamento e mutação mais conhecidas como operadores genéticos.

4.4.2 Seleção

A idéia principal do operador de seleção em um AG é oferecer aos indivíduos com maior probabilidade de aptidão, preferência para o processo de reprodução, permitindo que estes indivíduos possam passar as suas características às próximas gerações. O AG depende deste processo para evitar populações conhecidas como viciadas. Estas populações apontam sempre os mesmos indivíduos como os mais aptos à resolução do problema, criando populações sem grandes alterações genéticas.

A probabilidade de aptidão é dada pela aptidão (*Fitness*) calculada pela função objetivo do problema e dividida pelo somatório dos *Fitness* de todos os outros indivíduos da população:

$$\max imizar\ z = \sum_{i=1}^{m} l_i x_i$$

Modelo 3 – Função Objetivo PCU (Aptidão – *Fitness*).

A função é maximizar o somatório dos valores de utilidade l_i, valores que serão o próprio comprimento do item utilizado. A soma dos comprimentos dos itens utilizados não poderá ultrapassar a restrição física do PCU, que, neste caso, é o tamanho do objeto a ser cortado L (Modelo 4).

$$\max imizar\ z = \sum_{i=1}^{m} l_i x_i \leq L$$

Modelo 4 – Restrição física para o PCU.

Com esta função, o AG poderá determinar a aptidão (*fitness*) do cromossomo (indivíduo). O *fitness* é essencial para determinar a probabilidade que indicará qual indivíduo será mantido na evolução do algoritmo.

Indivíduos de baixa adequabilidade têm alta probabilidade de desaparecerem da população, ou seja, serem extintos, ao passo que indivíduos adequados terão grandes chances de sobreviverem. Esta forma de cálculo de aptidão é utilizada na seleção por roleta: veja exemplo abaixo [10]:

Cromossomo nº	String	Aptidão	% do total
1	0101101	45	13,2
2	1011001	89	26,2
3	1111101	125	36,7
4	0010101	21	6,1
5	0110100	52	15,2
6	0001001	9	2,6
	Total	341	100,0

Tabela 4 – Valores de exemplo para ilustrar seleção por roleta [10].

Com os valores percentuais constantes na quarta coluna da Tabela 4, a roleta constante da Figura 4 será elaborada. Esta roleta será girada 6 vezes para efetuar a seleção da população auxiliar (amostra) levando em conta que, os indivíduos com maior área na roleta tem, consequentemente, maiores chances de serem selecionados mais vezes do que os indivíduos menos aptos.

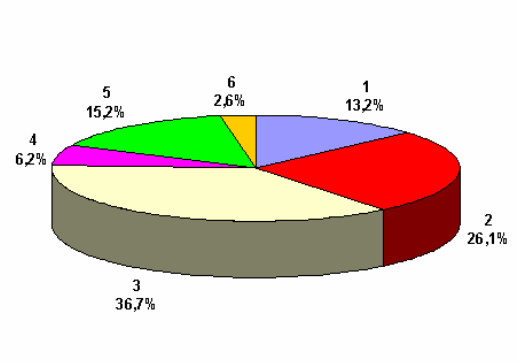

Figura 4 – Representação gráfica da roleta [10].

Como pôde ser visto nos modelos, são utilizados dois vetores. Um vetor binário x_i que assume valores 0 ou 1, que representa um indivíduo na população e consequentemente uma solução para o PCU. E um segundo vetor l_i que será os valores dos itens que o objeto poderá utilizar para realização do corte. Veja abaixo a representação cromossômica proposta e o um exemplo dos vetores x_i, l_i. Neste exemplo é utilizada um objeto com tamanho total L=150 cm.

0|1	0|1	0|1	0|1	0|1	0|1	0|1	0|1

Figura 5 – Representação cromossômica e vetor binário $\mathbf{x_i}$ alelos 0 ou 1 [13].

75	10	50	150	5	120	15	23

Figura 6 - Exemplo do vetor de itens $\mathbf{l_i}$ com valores entre 0 a **L** (Tamanho total da barra) [13].

Os dois vetores x_i e l_i deverão ter as mesmas dimensões do cromossomo, ou seja, terão o mesmo número de genes do cromossomo. Isso ocorre porque cada valor binário (alelo) determina se o item será utilizado na solução ou não. Caso o valor de x_i seja igual a 1, o seu referente item l_i está inserido na solução, caso contrário não. Veja que a representação do cromossomo será o próprio vetor binário x_i.

4.4.3 Cruzamento

Com a representação cromossômica e a forma de seleção determinadas, pode-se definir também outra forma de evolução para um indivíduo. A partir da seleção pode ser realizado o cruzamento dos indivíduos considerados mais aptos.

Cruzamento é a troca de fragmentos entre pares de cromossomos. Pode ser considerado como uma das principais características que diferenciam AG's de outras técnicas.

Na forma mais simples, trata-se de um processo aleatório que gera a propagação das características dos indivíduos mais aptos da população por meio de troca de segmentos e informações entre os mesmos, resultando em novos indivíduos. Veja abaixo o cruzamento, dentre vários, escolhido como ideal para o PCU. Cruzamento em apenas um ponto do cromossomo: é feita a escolha aleatória de um ponto onde será feita a troca de material cromossômico entre dois indivíduos (Figura 7).

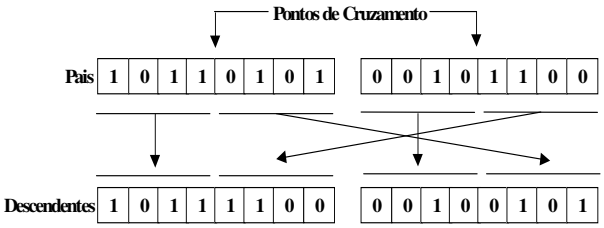

Figura 7 – Cruzamento de um ponto.

4.4.3 Cruzamento

Por último, tem-se o operador Mutação, que consiste na troca de um ou mais genes de um cromossomo. É equivalente a uma busca aleatória, sendo responsável por uma diferenciação do material genético dos indivíduos. Basicamente, seleciona-se uma posição num cromossomo e muda-se o valor do gene correspondente aleatoriamente para outro valor, por exemplo, trocar bit 0 por 1 ou 1 por 0.

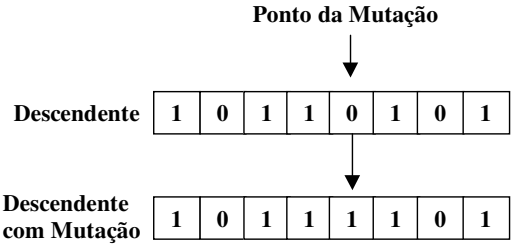

Figura 8 – Ocorrência de uma mutação.

Na Figura 9, é mostrado o pseudocódigo para um AG simples utilizado para implementação da resolução para o PCU.

```
Início
  t ← 0;
  Inicializar P(t);       //Cria uma população inicial.
  Avaliar P(t);           //Avalia a população criada
  Enquanto (não Condição_para_Término) faça
  Início
    t ← t + 1;
    Selecionar P(t) a partir de P(t - 1);
    Cruzar P(t);
    Mutar P(t);
    Avaliar P(t);
  Fim
Fim
```

Figura 9 – Algoritmo Genético padrão.

5. Resultados Experimentais

Como já foi visto anteriormente, foram quatro os algoritmos usados para a solução do PCU. O primeiro obstáculo encontrado foi a dificuldade de se resolver o problema para múltiplos objetos utilizando os algoritmos LD e AGU. Para tanto tiveram que sofrer modificações e aplicações de estratégias para a sua solução. Foram duas as estratégias utilizadas: Execução normal (EN) e Execução utilizando padrões de corte (PC).

Execução normal: Nesta execução o algoritmo é executado em sua forma normal, ou seja, os algoritmos que resolvem múltiplos objetos, como FFD, foram executados sem alterações, sendo apenas aplicado às restrições do problema. Já aos algoritmos AGU, LD e AG's, que em sua forma de execução normal resolvem ou problema ou objeto por vez, foi adicionando uma lista na qual são guardados os itens que já foram utilizados por outros objetos. Essa estratégia foi utilizada porque, resolvendo múltiplos objetos em apenas uma rodada, o resultado ficava prejudicado. Desta forma, resolvendo um objeto de cada vez, aumentou a complexidade mas melhorou o resultado. Esta estratégia necessita que os algoritmos sejam executados várias vezes até não possuir mais nenhum item em estoque a ser produzido. Fica claro que essa execução fica invisível para o usuário.

Execução utilizando padrões: Nesta execução é determinado o resultado considerado viável para depois repeti-lo até o término de itens que satisfazem de forma igual os itens produzidos no padrão viável encontrado. Segundo [9], chama-se padrão de corte a maneira como um objeto em estoque é cortado para produção dos itens demandados, ou seja, é encontrado um resultado e feito o maior número de cortes iguais que os itens oferecem.

A seguir são mostradas as tabelas de resultados com soluções do PCU gerados aleatoriamente. Os testes serão realizados com um estoque de 500, 2000 itens a serem produzidos para criação de um objeto de tamanho L = 150.

No caso dos AG, o número de itens em estoque reflete no tamanho do cromossomo que será utilizado para resolução dos AG. Sendo, respectivamente, 500 e 2000. O tamanho da geração utilizada é igual a 10. Cada população apresentará 30 indivíduos, ou seja, 30 soluções a cada geração. Isso resultará na criação de 300 soluções.

As tabelas abaixo apresentam o número de itens e a quantidade de objetos, que foram necessários para produzir estes itens, o número de recomeços para determinar a crescente da complexidade do algoritmo juntamente com a quantidade de trabalho, tempo (utilizando um computador Pentium III 1.1 com 512 MB) e a avaliação da validade do resultado, sendo que, o resultado não deverá ser apenas rápido com pouca complexidade, mas também minimizar as perdas e maximizar o aproveitamento total.

Abaixo, a descrição de cada linha da tabela que representa um parâmetro para análise e comparação dos algoritmos:

Itens: Itens de tamanhos menores em estoque a serem produzidos. Por causa de um problema encontrado na prática, foi estipulado um intervalo de L >= 1 cm e L <= 150 cm.

Objetos: Objetos necessários para produzir os itens menores em estoque. Tamanho dos objetos L = 150 cm. Estes objetos podem ser barras de ferro, vidros, tecidos todos cortados em apenas uma direção, somente na horizontal ou na vertical.

Recomeços: É o número de vezes que os algoritmos recomeçam a procura por uma solução por julgarem inviável a solução encontrada.

Quantidade de trabalho: É o número de testes e cortes realizados pela operação dominante, na tentativa de utilizar um item corrente não utilizado.

Tempo(m): Medido em minutos e segundos, o tempo é um parâmetro chave para determinar o aumento de complexidade de um algoritmo. Nos quadros abaixo, o aumento de tempo para a resolução do problema com entradas maiores é facilmente visualizado.

Valor máximo (%): Parâmetro que avalia, dentro das soluções, qual foi o topo de utilização dos objetos. Uma solução, por exemplo, pode apresentar uma perda considerada pequena, mas que deixou sobra na maioria dos objetos.

Perda (%): A perda é avaliada no montante do resultado, verificando o percentual de sobras obtidas nos objetos, lembrando que o objetivo principal ainda é o de encontrar algoritmos que maximizem o maior número de objetos minimizando a perda.

Total de aproveitamento (%): Com este parâmetro pode-se ter uma idéia de qual objeto a solução começou a ter sobra e avaliar junto aos outros parâmetros a eficiência e eficácia dos algoritmos.

Algoritmos	FFD	LD	AGU	AG's
Itens	500	500	500	500
Objetos	261	266	261	262
Recomeços	500	56001	261	1571
Qtd de trabalho	65371	3174830	107488	271984
Tempo(m,s)	0,020	0,55	0,010	2,64
Vlr máximo (%)	100	100	100	100
Perda (%)	3,39	4,83	3,01	3,38
Aproveitamento	96,61	95,17	99,34	96,62

Tabela 4 – Tabela com os resultados para execução dos algoritmos usando a EO com uma entrada pequena de 500 itens em estoque.

Algoritmos	LD(PC)	AGU(PC)	AG's(PC)
Itens	500	500	500
Objetos	266	261	261
Recomeços	29552	121	719
Qtd de trabalho	1755939	51162	121798
Tempo(m)	0,32	0,010	1,21
Vlr. máximo (%)	100	100	100
Perda (%)	4,83	3,01	3,01
Aproveitamento	95,17	99,34	99,34

Tabela 5 – Tabela com os resultados para execução dos algoritmos usando a estratégia de PC com uma entrada pequena de 500 itens em estoque.

Nos resultados para a entrada de 500 itens, apresentados na tabela 4 (estratégia utilizando EO) e tabela 5 (utilizando PC), não foi possível visualizar uma complexidade em questão do tempo, mas sim, um número elevado de testes realizados pela operação dominante que verifica cada item a ser utilizado na solução.

Nestes resultados percebeu-se que, utilizando PC, a porcentagem de perda não se alterou em grande escala, mas diminuiu a quantidade de recomeços e trabalho dos algoritmos.

Algoritmos	FFD	LD	AGU	AG's
Itens	2000	2000	2000	2000
Objetos	1032	1038	1032	1032
Recomeços	2000	598866	1032	6191
Qtd de trabalho	1049051	86857489	1617176	2276412
Tempo(m)	0,060	31,10	0,050	9,19
Vlr máximo (%)	100	100	100	100
Perda (%)	3,18	1,91	1,34	1,34
Aproveitamento	96,82	98,09	98,66	98,66

Tabela 6 – Tabela com os resultados para execução dos algoritmos usando a EO com uma entrada pequena de 2000 itens em estoque.

Algoritmos	LD(PC)	AGU(PC)	AG's(PC)
Itens	2000	2000	2000
Objetos	1038	1032	1032
Recomeços	79057	139	833
Qtd de trabalho	14838424	240559	298353
Tempo(m)	3,73	0,010	1,25
Vlr máximo (%)	100	100	100
Perda (%)	1,91	1,34	1,34
Aproveitamento	98,09	98,66	98,66

Tabela 7 – Tabela com os resultados para execução dos algoritmos usando a estratégia de PC com uma entrada pequena de 2000 itens em estoque.

Os resultados para 2000 itens, vistos na tabela 6, mostram um aumento considerável no tempo, recomeços e quantidade de trabalho na execução dos algoritmos LD e AG sendo classificados como NP. É analisada uma impossibilidade de utilização da estratégia EO, como por exemplo, em aplicações comerciais. Outro ponto negativo foi o fato destes algoritmos apresentarem pouca melhora nos resultados estatísticos comparando aos algoritmos FFD e AGU.

Uma estratégia que poderia ser utilizada para a possível melhora dos algoritmos seria a de abortar a busca depois de uma certa quantidade de trabalho, sendo que o resultado considerado viável poderia ser encontrado em uma determinada faixa de execução. Isso não quer dizer, porém, que o resultado seria o melhor, podendo neste caso, apresentar a mesma idéia de solução de valor maximal [12] dos algoritmos FFD e AGU, afetando o resultado do algoritmo.

Na tabela 7 os resultados já foram diferentes. A partir daí, foi possível verificar uma melhora nos parâmetros utilizando a estratégia com PC,

justificando assim a possível utilização destes algoritmos.

Com estes resultados pôde-se identificar que, mesmo levando em consideração o tempo de execução, os algoritmos AG e LD apresentam melhor eficácia. Não se pode afirmar que algoritmos com resultado de valor maximal sempre se comportarão desta maneira quando as entradas forem modificadas, como por exemplo, entradas de itens muito variados, nas quais a troca ou a insistência por uma procura mais elaborada não ocorre, podendo assim deixar a desejar.

Observando a resolução dos objetos, perda e aproveitamento, mesmo considerando o tempo de execução, conforme o aumento da complexidade devido ao tamanho da entrada, os algoritmos com melhores resultados foram os AG utilizando PC. Essa conclusão está relacionada, principalmente, na resolução dos objetos. Os AG apresentaram um salto e uma disparidade interessante. Enquanto a melhor solução, ou seja, os primeiros objetos resolvidos nos algoritmos FFD, AGU apresentaram sempre cortes básicos ou viciados como, por exemplo:

L = 150 cm;
Objeto = 1 item de 150;
Objeto = 1 item de 130 + 1 item de 20;
Objeto = 1 item de 145 + 1 item de 5;
Utilizando AG's, os resultados ficam em faixas bem diferentes como:
L = 150 cm;
Objeto = 1 item de 75 + 1 item de 25 + 1 item de 20 + 1 item de 15 + 1 item de 10 + 1 item 5;

Devemos considerar que, para cada tipo de situação e entrada de dados, pode haver diferentes comportamentos. Este tipo de resultado apresentado pelos AG demonstra um resultado menos viciado apontando para uma região de resultado bem mesclada. Isso mostra que os resultados com entradas mais complexas poderiam ser viáveis, diferentes dos algoritmos com valor maximal. Seguindo a linha do AG, os algoritmos de LD também mesclam o resultado, porém com menos saltos entre eles. Este foi o principal ponto no qual se pôde verificar as vantagens na utilização de AG e LD que apresentam diversas formas de solução viável para o PCU. Isso seria muito útil se, por exemplo, fosse aumentado o número de restrições para o problema. Podemos supor, por exemplo, que além de apresentar uma solução "ótima" ela precisa do maior número possível de itens utilizados. Já no caso do corte em demanda, como a aplicação do PC, seria possível identificar quais itens teriam preferência na realização do corte.

6. Conclusão

Como visto nos resultados, pôde-se identificar a vasta aplicação das idéias dos AG, sendo que sua teoria foi baseada principalmente na resolução de problemas de otimização e combinatória, teoria esta, que desperta o interesse em seu estudo por abordar idéias que são facilmente exemplificadas na natureza. Assim, o estudo torna-se atrativo e de fácil visualização para muitos pesquisadores da IC.

Nos resultados apresentados, comentários referentes a cada um foram feitos, mas pode-se destacar que o crescimento e a forma das entradas podem alterá-los, principalmente, se levarmos em consideração as tabelas com o comportamento do AG que se mostrou inviável no momento que o numero de itens aumenta.

Levando em consideração o problema do projeto de paredes, no qual, atualmente, utiliza-se o LD para resolver o corte unidimensional, houve uma surpresa por ficarem muito próximos dos resultados, de FFD e AGU. É provável que o motivo destaque-se pelo aumento das entradas e algumas alterações nas versões que já tinham sido testadas.

Com estes resultados pode-se pensar em disponibilizar na ferramenta a possibilidade do usuário escolher a forma de como será realizada o corte, deixando como trabalho futuro a continuação e o aprimoramento dos algoritmos aqui citados e os testes de outros para resolução do problema. Algo mais interessante ainda seria a utilização de um único algoritmo para resolver o corte uni e bidimensional, o qual, hoje, é feito com dois. Com esta idéia, também será possível ter uma noção, na prática, de qual algoritmo seria o ideal para o problema proposto.

Referências

[1] - COSTA, Ilaim. Jr. Introdução aos Algoritmos Genéticos. In: VII ESCOLA DE INFORMÁTICA DA SBC REGIONAL SUL, May 1999, Chapecó. **Anais...** SBC, 1999. 184p. p35-54.

[2] - GOLDBERG, E. David. **Genetic Algorithms in Search, Optimization and Machine Learnning**, Addison-Wesley, Reading, MA. 1989.

[3] - HOFF, Arild; LOKKETANGEN, Arne; MITTET, Ingvar. **Genetic Algorithms for 0/1 Multidimensional Knapsack Problems**. Disponível em: http://citeseer.ist.psu.edu/270864.html. Acesso em 02 jun. 2004.

[4] - HOLLAND, J.H. **Adaptation in natural and artificial systems**, Univ. of Michigan Press, AnnArbor.1975

[5] - KRÖGER, Berthold. Guillotinable bin packing: A genetic approach**. European Journal of Operational Researsh.** NO. 84, p.645-661. 1995.
[6] - KRÖGER, Berthold; SCHWENDERLING, Peter; VONBERGER, Oliver. **Parallel Genetic Packing of Rectangles.** Department os Mthematics and Computer Science, University of Osnabrück. Germany. 1995.
[7] - NEWTON, Charles; HOFFMAN, David**.** A New Approach to Cutting Stock Problems With and Without Contiguity. **Accepted by Computers and Operations Research.**
[8] - SERRADA, Anselmo P. **Una Introducción a la Computación Evolutiva.** Disponível por download em http://geocities.com/igoryepes. Acesso: (02 mar. 2001).
[9] - YANASSE, H.H.; FURTADO, J.C. at al**.** O Problema de Corte e Empacotamento e Aplicações Industriais. In: 2ª OFICINA NACIONAL DE PCE. XX CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL. 1997, Gramado. **Mini Curso...** Gramado: 1997. 145p.
[10] - YEPES, Igor. **Projeto Isis - Sistemas Inteligentes.** Disponível em: http://geocities.com/igoryepes. Acesso: 02 mar. 2001.
[11] - TANOMARU, J. Motivação, Fundamentos e Aplicações de Algoritmos Genéticos. In: II CONGRESSO BRASILEIRO DE REDES NEURAIS. III ESCOLA DE REDES NEURAIS. Out 1995, Curitiba. **Anais...** Curitiba: 1995 Copel.
[12] - TOSCANI, V. Laira; VELOSO, A. S. Paulo. **Complexidade de algoritmos.** 1. ed. Porto Alegre: Sagra Luzzatto, 2001. 202 p.
[13] - ZONTA, Tiago; COSTA, Ilaim. Jr. **Inteligência Computacional Aplicada à Resolução do Problema do Corte Unidimensional.** Monografia (Bacharel em ciência da computação) – Graduação Universidade do Oeste de Santa Catarina, Chapecó, 2001.
[14] GRAMANI, M. C. Minimizing the Number of Different Cutting patterns – a minimum path approach. In: **EURO/INFORMS JOINT INTERNATIONAL MEETING**. 2003, Istanbul, Turkey.
[15] – CINTRA, G.; WAKABAYASHI, Y. Dynamic Programming and Column Generation Based Approaches for Two-Dimensional Guillotine Cutting Problems, **Lecture Notes in Computer Science**, v. 3059, p. 175 – 190, 2004.